IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 16, 2019, accepted May 5, 2019, date of publication May 14, 2019, date of current version June 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2916828

Insights Into LSTM Fully Convolutional Networks for Time Series Classification

FAZLE KARIM[1], (Graduate Student Member, IEEE), SOMSHUBRA MAJUMDAR[2], AND HOUSHANG DARABI[1], (Senior Member, IEEE)

[1]Department of Mechanical and Industrial Engineering, University of Illinois at Chicago, Chicago, IL 60607, USA
[2]Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607, USA

Corresponding author: Houshang Darabi (hdarabi@uic.edu)

The Research Open Access Publishing (ROAAP) Fund of the University of Illinois at Chicago financially supported towards the open access publishing fee for this article.

ABSTRACT Long short-term memory fully convolutional neural networks (LSTM-FCNs) and Attention LSTM-FCN (ALSTM-FCN) have shown to achieve the state-of-the-art performance on the task of classifying time series signals on the old University of California-Riverside (UCR) time series repository. However, there has been no study on why LSTM-FCN and ALSTM-FCN perform well. In this paper, we perform a series of ablation tests (3627 experiments) on the LSTM-FCN and ALSTM-FCN to provide a better understanding of the model and each of its sub-modules. The results from the ablation tests on the ALSTM-FCN and LSTM-FCN show that the LSTM and the FCN blocks perform better when applied in a conjoined manner. Two z-normalizing techniques, z-normalizing each sample independently and z-normalizing the whole dataset, are compared using a Wilcoxson signed-rank test to show a statistical difference in performance. In addition, we provide an understanding of the impact dimension shuffle that has on LSTM-FCN by comparing its performance with LSTM-FCN when no dimension shuffle is applied. Finally, we demonstrate the performance of the LSTM-FCN when the LSTM block is replaced by a gated recurrent unit (GRU), basic neural network (RNN), and dense block.

INDEX TERMS Convolutional neural network, long short term memory recurrent neural network, time series classification.

I. INTRODUCTION

Time series classification has recently received a lot of attention over the past three decades [1]–[4]. Such data is widely available everywhere and collected with various sensors [5]. A variety of real world sensors capture time series information such as weather readings [6], stock market data [7], and EEG / ECG [8], [9].Time series classification is a supervised learning task that classifies a series of data points that are commonly collected in equal intervals and depicted in a sequential order [10]. Typically, the input to a time series classification problem is a time series signal, $X \in \mathbb{R}^{T \times F}$, such that $X_t \in \mathbb{R}^F$ is the input feature vector of length F at time step t, where $0 < t \leq T$. The maximum length of each time series, T, may vary [11]. The output of a time series classification problem, $Y \in \{1, \ldots, C\}$, is a discrete class/category label that represents the input time series signals. The total number of classes, C, is dependent on the time series classification problem. The main challenges faced in time series classification are how to efficiently (speed and space) [12] and effectively (accurately) [13] classify a time series.

Some of the earliest work that applies data mining techniques for time series classification dates back to the early 1990s when authors would apply various algorithms onto single artificial datasets [14], [15]. Since the initial decade of research in this field, Chen *et al.* [16] have graciously helped the community by collecting and making 85 time series datasets from various domains available online to the public for research purposes. This has lead to rapid progress in the field of time series classification and yielded a significant body of work. Recently, Dau *et al.* [17] have updated the repository with 43 datasets with time series datasets. These datasets have a significantly higher number of samples, several of which have long time dependencies or incorporates variable sequence lengths, which makes the task of sequence classification far more exigent. Most of the new datasets also have a significantly larger test set and a few have variable time series lengths to represent real-world scenarios [1].

The associate editor coordinating the review of this manuscript and approving it for publication was Xi Peng.

Several researchers have used the old archive benchmark datasets to propose feature-based models [18]–[22], ensembles [23], [24] and deep learning models [25]–[27] to accurately classify the time series data. One of the current state-of-the-art models that classify the time series datasets from the repository developed by Chen *et al.* [16] are the Long Short Term Memory Fully Convolutional Network (LSTM-FCN) and the Attention LSTM-FCN proposed by Karim *et al.* [27]. LSTM-FCN and ALSTM-FCN are deep learning models, a Fully Convolutional Network (FCN) module augmented with a Long Term Short Term Recurrent Neural Network (LSTM) that classify time series datasets. LSTM-FCN and ALSTM-FCN have received a lot of attention from the time series classification community due to their advantage over other models. In terms of classification accuracy, both the models outperform several traditional time series classification models, while requiring minimal pre-processing of the data. A significant advantage of utilizing these models is their ability to compute features on their own, eliminating the requirement for significant domain expertise and manual feature extraction. Furthermore, both these models can easily scale with a larger amount of time series data, which is generated daily by automated processes. Finally, LSTM-FCN has already been deployed in real world scenarios. One such application is to efficiently classify pet dog sounds using resource constrained sensors [28]. The original models, LSTM-FCN and ALSTM-FCN, lacked the explanation of each sub-module. In this paper, we provide detailed ablation tests to explain the sub-modules of the models.

The remainder of the paper is organized as follows. Section III presents the parameters used in developing the models and discusses the experiments performed. Section IV compares two *z-normalization* schemes. Subsequently, Section V provides a detailed ablation test on the deep learning models, LSTM-FCN and ALSTM. Finally, Section VI concludes the paper.

II. BACKGROUND REVIEW

A. TEMPORAL CONVOLUTIONS NETWORKS

Temporal convolution network is a type of artificial neural network whose input is generally a time series signal, X, where $X_t \in \mathbb{R}^F$ is the input feature vector of length F for time step t for $0 < t \leq T$. T may vary for each time series sequence [11].

In a temporal convolution network, 1D filters are applied on each convolutional layer, L, that discovers the evolution of the input signal over the course of an action. Lea *et al.* [11] discusses each filter of each layer are parameterized by tensor $W^{(l)} \in \mathbb{R}^{F_l \times d \times F_{l-1}}$ and biases $b^{(l)} \in \mathbb{R}^{F_l}$, where $l \in \{1, ..., L\}$ is the layer index and d is the filter duration. The i-th element of the activation $\hat{\mathbf{E}}_t^{(l)} \in \mathbb{R}^{F_l}$ of the l-th layer is a function of the activation matrix $E^{(l-1)} \in \mathbb{R}^{F_{l-1} \times T_{l-1}}$ of the previous layer, such that,

$$\hat{\mathbf{E}}_{i,t}^{(l)} = f\left(b_i^{(l)} + \sum_{t'=1}^{d} \left\langle W_{i,t',.}^{(l)}, E_{.,t+d-t'}^{(l-1)} \right\rangle\right) \tag{1}$$

for each time t where $f(\cdot)$ is a Rectified Linear Unit.

Typically, a convolutional layer is followed by batch normalization [29]. Subsequently, this is trailed by an activation function (a Rectified Linear Unit or a Parametric Rectified Linear Unit [30]).

B. RECURRENT NEURAL NETWORKS

Recurrent Neural Networks (RNN) are a type of artificial neural network that demonstrates stateful temporal behavior given a time sequence. Pascanu *et al.* [31] proposed an RNN to preserve a hidden vector $\mathbf{h}$ as a state that is updated at time step t,

$$\mathbf{h}_t = \tanh(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{I}\mathbf{x}_t), \tag{2}$$

where *tanh* is the hyperbolic tangent function, $\mathbf{x}_t$ is the input vector at time step t, $\mathbf{W}$ is the recurrent weight matrix and $\mathbf{I}$ is the projection matrix. The prediction, $\mathbf{y}_t$, is computed such that,

$$\mathbf{y}_t = \text{softmax}(\mathbf{W}\mathbf{h}_{t-1}), \tag{3}$$

where $\mathbf{h}$ is a hidden state, $\mathbf{W}$ is a weight matrix and softmax operation normalizes the output of the model to a valid probability distribution and the logistic sigmoid function is shown as σ. Deep RNNs can be formed by stacking the output of one RNN as the input to another, such that the hidden state, $\mathbf{h}^{l-1}$ of a RNN layer $l-1$, is an input to the hidden state, $\mathbf{h}^l$ of another RNN layer l. In other words,

$$\mathbf{h}_t^l = \sigma(\mathbf{W}\mathbf{h}_{t-1}^l + \mathbf{I}\mathbf{h}_t^{l-1}). \tag{4}$$

RNNs are prone to be affected by vanishing gradients. This issue is addressed using a Long short-term memory (LSTM) or a Gated Recurrent Unit (GRU).

C. LONG SHORT-TERM MEMORY RNNs

To solve the vanishing gradient problem, LSTM RNNs utilize gating functions in their state dynamics [32]. Each LSTM cell contains a hidden vector, $\mathbf{h}$, and a memory vector, $\mathbf{m}$. At each time step, the memory vector regulates the state updates and outputs, such that the following computation is performed computed as follows (first depicted by Graves *et al.* [33]):

$$\begin{aligned}
\mathbf{g}^u &= \sigma(\mathbf{W}^u\mathbf{h}_{t-1} + \mathbf{I}^u\mathbf{x}_t) \\
\mathbf{g}^f &= \sigma(\mathbf{W}^f\mathbf{h}_{t-1} + \mathbf{I}^f\mathbf{x}_t) \\
\mathbf{g}^o &= \sigma(\mathbf{W}^o\mathbf{h}_{t-1} + \mathbf{I}^o\mathbf{x}_t) \\
\mathbf{g}^c &= \tanh(\mathbf{W}^c\mathbf{h}_{t-1} + \mathbf{I}^c\mathbf{x}_t) \\
\mathbf{m}_t &= \mathbf{g}^f \odot \mathbf{m}_{t-1} + \mathbf{g}^u \odot \mathbf{g}^c \\
\mathbf{h}_t &= \tanh(\mathbf{g}^o \odot \mathbf{m}_t)
\end{aligned} \tag{5}$$

where $\mathbf{g}^u, \mathbf{g}^f, \mathbf{g}^o, \mathbf{g}^c$ are the activation vectors of the input, forget, output and cell state gates respectively, $\mathbf{W}^u, \mathbf{W}^f, \mathbf{W}^o, \mathbf{W}^c$ are the recurrent weight matrices, $\mathbf{I}^u, \mathbf{I}^f, \mathbf{I}^o, \mathbf{I}^c$ portrays the projection matrices, σ is the logistic sigmoid function, $\odot$ is an elementwise multiplication, and h_t is the hidden state vector of the tth time step.

D. GATED RECURRENT UNIT

Cho *et al.* [34] proposed a modification to the LSTM RNN that also solves the vanishing gradient problem using an update and reset gate. Due to the simpler gating structure of the model, reduced number of gates and thereby parameters, it is considered to be an efficient alternative to the LSTM RNN.

$$\mathbf{z_t} = \sigma_g(\mathbf{W_z x_t} + \mathbf{U_z h_{t-1}}) \tag{6}$$

$$\mathbf{r_t} = \sigma_g(\mathbf{W_r x_t} + \mathbf{U_r h_{t-1}}) \tag{7}$$

$$\mathbf{h_t} = (1 - \mathbf{z_t}) \odot \mathbf{h_{t-1}} + \mathbf{z_t} \odot \sigma_h(\mathbf{W_h x_t} + \mathbf{U_h}(\mathbf{r_t} \odot \mathbf{h_{t-1}})) \tag{8}$$

where x_t is the input vector at time step t, z_t is the update gate vector, r_t is the reset gate vector, h_t is the hidden state and output vector, W_z and W_r are the trainable weight matrices for the update and reset gate respectively, U_z and U_r are the trainable recurrent weight matrices for the update and reset gate respectively, σ_g is the logistic sigmoid function, σ_h is the hyperbolic tangent function and $\odot$ is the Hadamard product of the two inputs.

E. FULLY CONNECTED (DENSE) LAYER

A fully connected layer can be described as a dense matrix multiplication of the input vector with a trainable weight matrix, and optionally, the addition of a trainable bias vector to the output. The output of each layer can be represented by:

$$output = a(\mathbf{W}\mathbf{x} + b) \tag{9}$$

where $\mathbf{W}$ is a weight matrix, b is a bias vector, and a is a non-linear activation function. Common activation functions are the Rectified Linear Unit (ReLU), the logistic sigmoid function, or a hyperbolic tangent function.

III. EXPERIMENTS

The LSTM-FCN and ALSTM-FCN models are trained on various released UCR benchmark datasets. The benchmark datasets include a train and test set which is used for model training and validation. We utilize the same structure of the models as the original models [27] and perform grid search to find the optimal number of LSTM cells from the set consisting of 8, 64 or 128 cells. All models are trained for 2000 epochs. The batch size of 128 is kept consistent for all datasets. All LSTM or Attention LSTM layers are followed by dropout layer with a probability of 80 percent to prevent overfitting. Class imbalance is handled via a class weighing scheme inspired by King and Zeng [35]. All models are trained using the Keras library [36] with Tensorflow [37] as the backend and are made available publically.[1]

All models are trained via gradient descent using the Adam optimizer [38]. The initial learning rate was set to $1e\text{-}3$ and is reduced to a minimum of $1e\text{-}4$. We reduced the learning rate by a factor of $1/\sqrt[3]{2}$, whenever the training loss of 100 consecutive epochs do not improve. The model weights are updated only through the training loss. The accuracies we report are based on the best models we find. The methodology we follow is common in various deep learning applications [39]–[43]. In addition, we utilize the initialization proposed by He *et al.* [40] for all convolutional layers. The input data is *z-normalized* and the datasets with variable length time series are padded with zeros at the end to match the longest time series in that dataset. All models are evaluated using classification accuracy and *mean-per-class-error* (MPCE), which is defined as the average error of each class for all the datasets and mathematically represented as following:

$$PCE_k = \frac{1 - accuracy}{number\ of\ unique\ classes}$$

$$MPCE = \frac{1}{N}\sum_{k=1}^{N} PCE_k.$$

[1]The codes and weights of each model are made available at https://github.com/houshd/LSTM-FCN-Ablation

TABLE 1. Performance comparison of LSTM-FCN and ALSTM-FCN with the baseline models. Green cells designate instances where our performance matches or exceeds state-of-the-art results. Bold values denote model with the best performance.

Name	Baseline [1]	LSTM-FCN Data Normalized	ALSTM-FCN Data Normalized	LSTM-FCN Sample Normalized	ALSTM-FCN Sample Normalized
ACSF1	0.6400	**0.9300**	0.9100	0.9200	0.9200
AllGestWiX	0.7171	**0.7214**	0.7200	0.7071	0.7086
AllGestWiY	0.7300	0.7786	0.7914	**0.7929**	0.7829
AllGestWiZ	0.6514	**0.7400**	0.7357	0.6800	0.6914
BME	0.9800	**1.0000**	0.8333	0.9933	0.8600
Chinatown	0.9565	**0.9855**	**0.9855**	0.9826	0.9797
Crop	0.7117	**0.7652**	0.7638	0.7425	0.7389
DodgLpDay	0.5875	**0.6375**	0.4875	0.6125	0.5875
DodgLpGm	**0.9275**	0.8913	0.7754	0.8986	0.8261
DodgLpWnd	0.9855	**0.9855**	0.9710	0.9783	0.9275
EOGHzSgn	0.5028	0.6547	**0.6878**	0.6409	0.6133
EOGVtSgn	0.4751	**0.5387**	0.5138	0.5028	0.4696
EthLevel	0.2820	**0.7660**	0.7380	**0.7660**	0.7480
FrzRegTr	0.9070	0.9986	**0.9989**	**0.9989**	**0.9989**
FrzSmlTr	0.7533	0.8295	**0.8747**	0.8025	0.8407
Fungi	0.8387	**1.0000**	0.9946	0.9892	0.9839
GestMidAirD1	0.6385	**0.7462**	0.7154	0.7308	0.7231
GestMidAirD2	0.6077	0.6923	0.6385	**0.7077**	0.6923
GestMidAirD3	0.3769	**0.4538**	0.4462	**0.4538**	0.3846
GestPebZ1	0.8256	**0.9419**	0.9244	**0.9419**	0.9128
GestPebZ2	0.7785	**0.8987**	0.8354	0.8544	0.8861
GunPtAgeSp	0.9652	**1.0000**	**1.0000**	0.9968	**1.0000**
GunPointMVsF	0.9968	**1.0000**	**1.0000**	**1.0000**	**1.0000**
GunPointOVsY	0.9651	**1.0000**	**1.0000**	0.9968	0.9968
HouseTwenty	0.9412	0.9664	0.9496	**0.9832**	**0.9832**
InsEPGRegTr	0.8715	**1.0000**	**1.0000**	0.9960	**1.0000**
InsEPGSmlTr	0.7349	**1.0000**	**1.0000**	0.9478	0.9438
MelbPed	0.8482	0.9747	**0.9755**	0.9147	0.9135
MxShpRegTr	0.9089	**0.9748**	0.9720	0.9711	0.9678
MxShpSmlTr	0.8355	0.9365	0.9274	**0.9390**	0.9225
PickGestWiZ	0.6600	**0.9000**	**0.9000**	0.7600	0.7000
PigAryPress	0.1058	**0.7885**	0.7596	0.4231	0.3942
PigArtPress	0.2452	0.9808	0.9904	**1.0000**	**1.0000**
PigCVP	0.1587	0.9231	**0.9279**	0.8702	0.8702
PLAID	0.8399	0.4842	0.5047	**0.9088**	0.8994
PowerCons	0.9333	**1.0000**	0.9500	0.9944	0.9167
Rock	0.8400	0.8000	0.8000	**0.9200**	0.8600
SgHdGendCh2	0.8450	**0.9400**	0.8567	0.9200	0.8567
SgHdMovCh2	0.6378	**0.7044**	0.5622	0.5422	0.5556
SgHdSubCh2	0.8000	**0.9222**	0.6533	0.8800	0.7911
ShkGestWiZ	0.8600	**0.9800**	**0.9800**	0.9000	0.8800
SmthSub	0.9467	**1.0000**	**1.0000**	0.9867	0.9867
UMD	0.9931	**1.0000**	0.9931	0.9792	0.9722
MPCE	-	0.0172	0.0242	0.0191	0.0242
Count	1	30	14	12	6

IV. DATASET ABLATION TEST

Table 1 represents the accuracies obtained by applying LSTM-FCN and ALSTM-FCN on the 43 new UCR benchmark datasets based on two *z-normalization* schemes when normalizing the datasets prior to training. These 43 UCR benchmark datasets are the only datasets in the repository that are not padded, normalized or pre-processed in any way. The *dataset mean and standard deviation* is calculated as

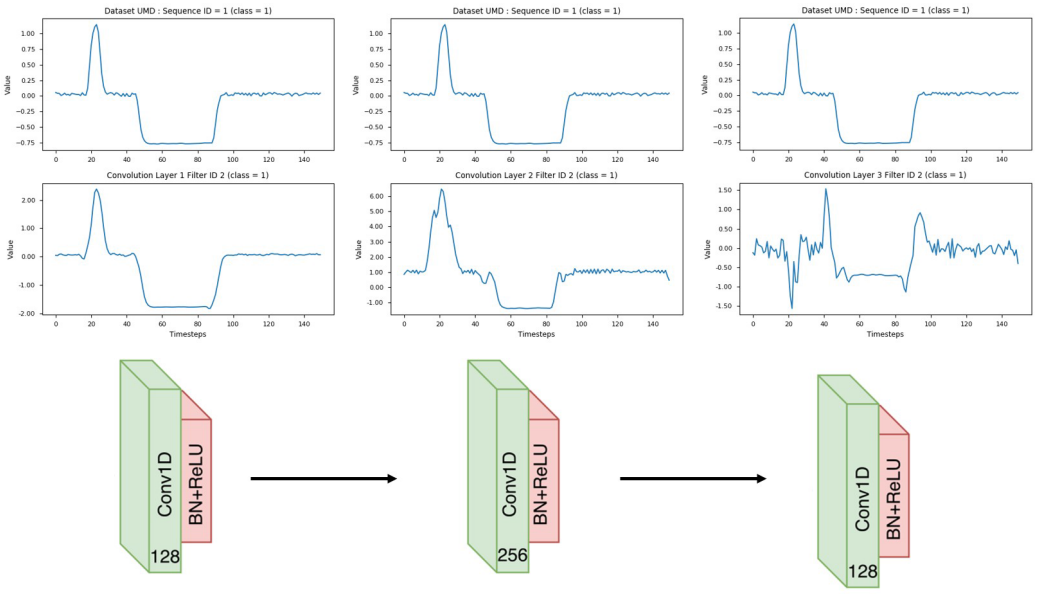


FIGURE 1. Ablation test–visual representation of the input signal after transformation through randomly selected filters from each convolutional layer.

the mean and standard deviation of only the train set, and then applied to both train and tests, whereas the *sample mean and standard deviation* was calculated for each individual sample separately. When using LSTM-FCN and ALSTM-FCN, our results indicate that when the whole dataset is *z-normalized*, it performs better on 34 datasets (LSTM-FCN) and 30 datasets (ALSTM-FCN) than when each sample is *z-normalized* separately. In addition, a Wilcoxon signed-rank test [44] was performed to compare this, yielding a p-value of 4.57*e-07*. We chose the significance level (alpha) of 0.05 for all statistical tests. Since the p-value is less than the *Dunn-Sidak* [45] corrected significance level (alpha) of 0.025, we conclude that *z-normalizing* the whole dataset performs differently than when *z-normalizing* each sample.

We recommend *z-normalizing* the whole dataset *iff* one knows that the train set can sufficiently represent the global population of the dataset. In other words, if no *a priori* information or domain knowledge is known about the train set, it is safer to *z-normalize* each sample separately, as explained by Dau *et al.* [1]. They provide an example explaining why it is safer to *z-normalize* each sample separately using the dataset *GunPoint*, where a video is converted into a time series. If another video is taken where ''the camera is zoomed in or out, or the actors stood a little closer to the camera, or that the female actor decided to wear new shoes with a high heel'' [1], the converted time series will be different. The train set will not have this distribution as the validation or test set, and the prediction made by this classifier will be off. In this scenario, it would be best to *z-normalize* each sample separately. On the other hand, if a domain expert knows the train set contains a wide range of samples that represent the different types and amplitudes of time series, *z-normalizing* via the dataset mean and standard deviation would be wiser when using LSTM-FCN and ALSTM-FCN as classifiers.

V. MODEL ABLATION TESTS

We perform an ablation study on our model to provide an understanding of the impact of each layer of our model and show how significantly they affect the performance measure. The LSTM-FCN and ALSTM-FCN models are applied to 61 datasets from the UCR repository, such that each dataset is sample *z-normalized*. Each dataset chosen were datasets that outperform the SOTA non-ensemble classifiers, BOSS [20] and WEASEL [46]. We apply BOSS and WEASEL on all UCR datasets based on code and default parameters provided by the author online. It should be noted, this paper is not comparing results with BOSS and WEASEL. BOSS and WEASEL is only used to select datasets that would provide a better understanding of LSTM-FCN and ALSTM-FCN when it performs well.

In addition, the significance level (alpha) of 0.05 is selected for all statistical tests. The null hypothesis and alternative hypothesis of all Wilcoxon signed-rank test are as follows:

$$H_o : Median_{proposed\ model} = Median_{compared\ model}$$
$$H_a : Median_{proposed\ model} \neq Median_{compared\ model}.$$

An essential point of discussion concerning the working of the LSTM-FCN and ALSTM-FCN model is the choice of utilizing an LSTM Recurrent module in conjunction with the FCN block. In the following ablation tests, we study the performance of the individual components which constitute the LSTM-FCN and ALSTM-FCN models, their performance compared to a linear baseline, as well as the empirical and statistical analysis on the performance of the individual components and the final model.

A. FULLY CONVOLUTIONAL BLOCK

LSTM-FCN and ALSTM-FCN comprise of a fully convolutional block and an LSTM/Attention LSTM block. The FCN

block has three stacked temporal convolutional blocks with the number of filters defined as 128, 256, and 128. Figure 1 depicts a visual representation of a single sample from the *UMD* dataset after transformation via a random filter selected from each of the convolutional blocks.

As can be noticed, a randomly selected filter from the first CNN block is applying a form of noise reduction that is learned via gradient descent, whereas two subsequent randomly selected filters from the later layers are transforming the data to be far more inconsistent. Based on our analysis of a few filters on various datasets, we conclude that the CNN filters in all layers act as feature extractors and transform the data into separable classes. The model learns the parameters of these transformations on its own via stochastic gradient descent. If a dataset sample requires the removal of noise, it is learned by a few filters of the first CNN layer. It is challenging to postulate what type of transformation is occurring in each filter, as the model transforms the data differently for each of the datasets, on the basis of random initialization of the convolution kernels and order of stochastic gradient descent updates. However, the filter parameters are learned such that their objective is to transform the data into separable classes.

In order to empirically demonstrate that the LSTM-FCN and ALSTM-FCN models are learning to separate the classes better, we examine the features from the FCN block by applying them to a tuned linear SVM classifier. The results are summarized in Table 2. The linear SVM classifier that is applied on the features extracted from the FCN block is better in 41 datasets (LSTM-FCN model) and 45 datasets (ALSTM-FCN model) as compared to when the tuned linear SVM classifier is applied on to the raw signal. Based on this knowledge, we conclude that the FCN block is transforming the data into separable classes.

B. LSTM/ALSTM RECURRENT BLOCK

Due to the dimensional shuffle that is applied before the LSTM block, the features extracted by LSTM block by itself do not contribute significantly to the overall performance. When these features are applied onto a tuned linear SVM classifier, the classifier is better in only 19 datasets (for the LSTM block) and 4 datasets (for the ALSTM block) as compared to when the tuned linear SVM classifier is applied to the raw input dataset. The above indicates that the LSTM, by itself, is not separating the data into linear separable classes.

C. LSTM/ALSTM CONCATENATED WITH FCN BLOCK

Nevertheless, when the features of the LSTM block/ALSTM block are concatenated with the CNN features, we obtain a more robust set of features that can better separate the classes of the dataset. The above insight is statistically validated by applying the concatenated features to a single layer perceptron classifier which accepts the extracted features as input (due to the fact that the data is transformed into separable classes). The training scheme of all perceptron models is kept consistent with how we train all LSTM-FCN and ALSTM-FCN models, as detailed in Section III. Results, shown in Table 3, show that the features from of the LSTM/ALSTM block coupled with the features from the FCN block improve the model performance.

For the ALSTM-FCN model, the ALSTM features joined with the FCN features outperform the features from the ALSTM block or the FCN block on 49 datasets, yielding to a p-value of $1.34e\text{-}08$ when a Wilcoxon Signed-rank test [44] is applied. Similarly, the LSTM features joined with the FCN features in the model LSTM-FCN outperform the features from the LSTM block or the FCN block on 54 datasets, yielding to a p-value of $1.22e\text{-}08$. The *Dunn-Sidak* [45] corrected significant alpha value is 0.02.

TABLE 2. Ablation test - linear SVM performance comparison of LSTM/ALSTM Block, FCN Block with the raw signals. Green cells and orange cells designate instances where the linear SVM model on the block exceeds the linear SVM on raw signals. Bold values denotes the block with the best performance using the linear SVM classifier. Count* represents the number of bold values in that column.

Dataset	Raw	FCN Block	LSTM Block	Raw	FCN Block	ALSTM Block
Car	**0.83**	0.42	0.78	**0.83**	0.23	0.22
ChlConc	0.57	0.53	**0.61**	**0.57**	0.53	0.53
Compt	0.52	0.50	**0.54**	0.52	**0.77**	0.50
Cricket_X	0.27	**0.36**	0.33	0.27	**0.45**	0.12
Cricket_Z	0.28	0.51	**0.54**	0.28	**0.67**	0.09
DiaSzRed	**0.94**	0.31	0.93	**0.94**	0.31	0.30
DsPhOutAgGp	0.80	0.78	**0.81**	0.80	**0.82**	0.64
DsPhxOCor	0.53	**0.81**	0.48	0.53	**0.63**	**0.63**
DsPhxTW	0.76	0.74	**0.77**	0.76	**0.79**	0.53
Earthquakes	0.57	**0.82**	0.71	0.57	**0.82**	0.78
FaceAll	0.68	**0.92**	0.77	0.68	**0.95**	0.18
FordB	0.49	**0.88**	0.56	0.49	0.49	**0.50**
Ham	**0.70**	0.51	0.66	**0.70**	0.51	0.51
Haptics	**0.44**	0.19	0.41	**0.44**	0.19	0.21
ItyPwrDmd	0.96	0.97	**0.97**	0.96	**0.96**	0.50
LgKchApp	0.39	**0.52**	0.40	0.39	**0.68**	0.33
Lighting7	0.64	**0.77**	0.59	0.64	**0.68**	0.26
Mallat	**0.88**	0.12	0.54	**0.88**	0.12	0.12
MedImg	0.56	**0.77**	0.56	0.56	**0.76**	0.51
MidPhxOtAgGrp	**0.80**	0.75	0.80	**0.80**	0.76	0.27
MidPhxOtCor	0.53	**0.82**	0.55	0.53	**0.65**	**0.65**
MidPhxTW	0.64	0.60	**0.65**	**0.64**	0.61	0.21
NonECG_Thor1	**0.91**	0.19	0.85	**0.91**	0.22	0.02
NonECG_Th2	**0.92**	0.17	0.20	**0.92**	0.22	0.02
OSULeaf	0.42	**0.48**	0.41	0.42	**0.50**	0.18
PhgOtCor	0.66	**0.82**	0.66	**0.66**	0.61	0.61
PrxPhxOtAgeGp	**0.85**	0.84	0.84	**0.85**	0.84	0.49
PrxPhxOtCor	**0.79**	0.68	0.75	0.79	**0.91**	0.68
PrxPhxTW	0.79	**0.81**	0.65	0.79	**0.83**	0.45
ScreenType	**0.38**	0.33	0.37	0.38	**0.51**	0.33
SonyAIBO	0.66	**0.95**	0.64	0.66	**0.99**	0.43
SonyAIBOII	0.81	**0.90**	0.82	0.81	**0.88**	0.44
SwdLeaf	0.79	**0.97**	0.81	0.79	**0.98**	0.05
Symbols	0.79	**0.92**	0.82	0.79	**0.81**	0.17
ToeSeg1	0.56	**0.97**	0.56	0.56	**0.86**	0.55
TwoLeadECG	0.89	**0.99**	0.64	0.89	**1.00**	0.50
ACSF1	0.59	**0.92**	0.33	0.59	**0.88**	0.10
AllGestWiX	0.27	**0.67**	0.30	0.27	**0.66**	0.10
AllGestWiY	0.35	**0.74**	0.30	0.35	**0.73**	0.11
AllGestWiZ	0.30	**0.65**	0.24	0.30	**0.65**	0.10
Chinatown	0.97	**0.98**	0.91	0.97	**0.98**	0.72
Crop	0.69	**0.72**	0.45	0.69	**0.72**	0.34
EOGHzSgn	0.43	**0.53**	0.24	0.43	**0.55**	0.08
EOGVtSgn	0.35	**0.42**	0.29	0.35	**0.35**	0.08
EthLevel	**0.75**	0.75	0.25	**0.75**	0.71	0.25
FrzRegTr	0.98	**1.00**	0.80	0.98	**1.00**	0.50
GestPebZ1	0.72	**0.85**	0.69	0.72	**0.87**	0.22
GunPointMVsF	0.98	**1.00**	0.89	0.98	**1.00**	0.53
GunPointOVsY	0.88	**0.97**	0.86	0.88	**0.97**	0.48
InsEPGRegTr	0.64	**1.00**	0.59	0.64	**1.00**	0.47
MelbPed	0.84	**0.91**	0.75	0.84	**0.90**	0.15
MxShpRegTr	0.81	**0.95**	0.80	0.81	**0.96**	0.17
MxShpSmlTr	0.80	**0.91**	0.81	0.80	**0.92**	0.13
PickGestWiZ	0.60	**0.70**	0.50	0.60	**0.68**	0.10
PigAryPress	0.06	**0.35**	0.02	0.06	**0.37**	0.02
PowerCons	**0.93**	0.89	0.83	**0.93**	0.87	0.50
SgHdGendCh2	**0.88**	0.82	0.81	**0.88**	0.77	0.35
SgHdMovCh2	0.48	**0.51**	0.35	0.48	**0.51**	0.17
ShkGestWiZ	0.62	**0.86**	0.36	0.62	**0.86**	0.10
SmthSub	0.67	**0.97**	0.61	0.67	**0.96**	0.33
UMD	**0.98**	0.97	0.71	0.98	**0.99**	0.22
Count	15	38	7	15	46	3

TABLE 3. Ablation test - MLP performance comparison of LSTM/ALSTM Block, FCN Block, LSTM/ALSTM-FCN Block and the raw signals. Green cells and orange cells designate instances where the MLP model on the block exceeds the MLP on raw signals. Bold values denotes the block with the best performance using the MLP classifier. Count* represents the number of bold values in that column.

Dataset	Raw	FCN Block	LSTM Block	LSTM-FCN	Raw	FCN Block	ALSTM Block	ALSTM-FCN
Car	0.83	0.42	0.78	**0.95**	0.83	0.23	0.22	**0.92**
ChlConc	0.57	0.53	0.61	**0.80**	0.57	0.53	0.53	**0.79**
Compt	0.52	0.50	0.54	**0.84**	0.52	0.77	0.50	**0.84**
Cricket_X	0.27	0.36	0.33	**0.78**	0.27	0.45	0.12	**0.78**
Cricket_Z	0.28	0.51	0.54	**0.82**	0.28	0.67	0.09	**0.79**
DiaSzRed	**0.94**	0.31	0.93	**0.94**	**0.94**	0.31	0.30	0.94
DsPhOutAgGp	0.80	0.78	0.81	**0.83**	0.80	0.82	0.64	**0.83**
DsPhxOCor	0.53	0.81	0.48	**0.81**	0.53	0.63	0.63	**0.81**
DsPhxTW	0.76	0.74	0.77	**0.79**	0.76	0.79	0.53	**0.79**
Earthquakes	0.57	**0.82**	0.71	0.80	0.57	**0.82**	0.78	0.80
FaceAll	0.68	0.92	0.77	**0.92**	0.68	**0.95**	0.18	0.93
FordB	0.49	0.88	0.56	**0.89**	0.49	0.49	0.50	**0.88**
Ham	0.70	0.51	0.66	**0.73**	0.70	0.51	0.51	**0.72**
Haptics	0.44	0.19	0.41	**0.49**	0.44	0.19	0.21	**0.50**
ItyPwrDmd	0.96	0.97	**0.97**	0.96	0.96	**0.96**	0.50	**0.96**
LgKchApp	0.39	0.52	0.40	**0.89**	0.39	0.68	0.33	**0.91**
Lighting7	0.64	0.77	0.59	**0.79**	0.64	0.68	0.26	**0.77**
Mallat	0.88	0.12	0.54	**0.97**	0.88	0.12	0.12	**0.97**
MedImg	0.56	0.77	0.56	**0.79**	0.56	0.76	0.51	**0.77**
MidPhxOtAgGrp	**0.80**	0.75	0.80	0.75	**0.80**	0.76	0.27	0.76
MidPhxOtCor	0.53	0.82	0.55	**0.83**	0.53	0.65	0.65	**0.83**
MidPhxTW	0.64	0.60	**0.65**	0.61	**0.64**	0.61	0.21	0.61
NonECG_Thor1	0.91	0.19	0.85	**0.96**	0.91	0.22	0.02	**0.96**
NonECG_Th2	0.92	0.17	0.20	**0.96**	0.92	0.22	0.02	**0.95**
OSULeaf	0.42	0.48	0.41	**0.98**	0.42	0.50	0.18	**0.99**
PhgOtCor	0.66	0.82	0.66	**0.83**	0.66	0.61	0.61	**0.83**
PrxPhxOtAgeGp	0.85	0.84	0.84	**0.87**	0.85	0.84	0.49	**0.86**
PrxPhxOtCor	0.79	0.68	0.75	**0.91**	0.79	0.91	0.68	**0.92**
PrxPhxTW	0.79	0.81	0.65	**0.82**	0.79	**0.83**	0.45	0.82
ScreenType	0.38	0.33	0.37	**0.62**	0.38	0.51	0.33	**0.64**
SonyAIBO	0.66	0.95	0.64	**0.96**	0.66	**0.99**	0.43	0.97
SonyAIBOII	0.81	0.90	0.82	**0.97**	0.81	0.88	0.44	**0.98**
SwdLeaf	0.79	0.97	0.81	**0.98**	0.79	**0.98**	0.05	0.97
Symbols	0.79	0.92	0.82	**0.98**	0.79	0.81	0.17	**0.97**
ToeSeg1	0.56	0.97	0.56	**0.98**	0.56	0.86	0.55	**0.99**
TwoLeadECG	0.89	0.99	0.64	**1.00**	0.89	1.00	0.50	**1.00**
ACSF1	0.40	**0.91**	0.36	**0.91**	0.40	0.89	0.10	**0.90**
AllGestWiX	0.24	0.66	0.30	**0.71**	0.24	0.65	0.10	**0.71**
AllGestWiY	0.32	0.72	0.29	**0.79**	0.32	0.73	0.10	**0.77**
AllGestWiZ	0.26	0.63	0.25	**0.68**	0.25	0.64	0.10	**0.68**
Chinatown	0.96	**0.98**	0.94	**0.98**	0.96	**0.98**	0.72	**0.98**
Crop	0.63	0.72	0.42	**0.74**	0.63	0.71	0.34	**0.74**
EOGHzSgn	0.30	0.54	0.27	**0.62**	0.30	0.55	0.08	**0.60**
EOGVtSgn	0.32	0.36	0.28	**0.49**	0.31	0.35	0.08	**0.45**
EthLevel	0.46	0.62	0.25	**0.68**	0.53	0.65	0.25	**0.73**
FrzRegTr	0.82	1.00	0.78	**1.00**	0.80	1.00	0.50	**1.00**
GestPebZ1	0.73	0.81	0.70	**0.94**	0.71	0.84	0.18	**0.91**
GunPointMVsF	0.85	1.00	0.88	**1.00**	0.84	1.00	0.53	**1.00**
GunPointOVsY	0.88	0.97	0.84	**1.00**	0.85	0.97	0.48	**0.99**
InsEPGRegTr	0.65	0.98	0.62	**1.00**	0.65	0.98	0.47	**1.00**
MelbPed	0.79	0.91	0.74	**0.91**	0.78	0.90	0.13	**0.91**
MxShpRegTr	0.80	0.95	0.80	**0.97**	0.81	0.96	0.24	**0.97**
MxShpSmlTr	0.75	0.92	0.80	**0.94**	0.75	0.92	0.19	**0.92**
PickGestWiZ	0.52	**0.70**	0.52	**0.70**	0.52	0.58	0.10	**0.64**
PigAryPress	0.05	0.39	0.01	**0.42**	0.05	**0.38**	0.02	0.38
PowerCons	0.89	0.87	0.82	**0.93**	0.89	0.88	0.50	**0.91**
SgHdGendCh2	0.75	0.81	0.80	**0.92**	0.76	0.79	0.65	**0.86**
SgHdMovCh2	0.38	0.46	0.34	**0.54**	0.38	0.48	0.17	**0.56**
ShkGestWiZ	0.44	0.86	0.30	**0.90**	0.44	**0.88**	0.10	**0.88**
SmthSub	0.69	0.97	0.57	**0.99**	0.70	0.95	0.33	**0.98**
UMD	0.88	0.94	0.73	**0.97**	0.88	**0.98**	0.22	0.97
Count	2	4	2	57	3	10	0	51

It is evident that when applying the LSTM block (with dimension shuffle) and the FCN block parallelly, the blocks augment each other, and force each other to detect a set of features which when combined, yield an overall better performing model. In other words, the LSTM block attached with the FCN block statistically helps improve the overall performance of the model providing informative features that in conjunction with the FCN features, are useful in separating the classes further.

D. DIMENSION SHUFFLE vs NO DIMENSION SHUFFLE

Another ablation test performed is to check the impact dimension shuffle has on the overall behavior of the model. The dimension shuffle transposes the input univariate time series of N time steps and 1 variable into a multivariate time series of N variables and 1 time step. In other words, when dimension shuffle is applied to the input before the LSTM block, the LSTM block will process only 1 time step with N variables.

In this ablation test, LSTM-FCN with dimension shuffle is compared to LSTM-FCN without dimension shuffle on all 128 UCR datasets using a cell size of 8, 64, 128 (yielding to a total of $128 \times 3 = 384$ experiments). LSTM-FCN with dimension shuffle outperforms LSTM-FCN without dimension shuffle on 258 experiments, ties in 27 experiments, and performs worse in 99 experiments. For the experiments when LSTM-FCN with dimension shuffle outperforms LSTM-FCN without dimension shuffle, the accuracy improved on average by 6.00%. Conversely, for the experiments when LSTM-FCN with dimension shuffle performs worse than LSTM-FCN without dimension shuffle, the accuracy is worse by an average of 5.26%. A Wilcoxson signed-rank test results in a p-value of $3.69E-17$, indicating a statistical difference in performance where LSTM-FCN with dimension shuffle performs better. This result is contrary to what most people would hypothesize. LSTM-FCN without dimension shuffle overfits the UCR datasets in more instances than LSTM-FCN with dimension shuffle. This is because the LSTM block without dimension shuffle by itself performs extremely well. The FCN block and LSTM block without the dimension shuffle does not benefit each other.

Another critical fact to note is that the LSTM-FCN with dimension shuffle processes the univariate time series in one time step. The gating mechanisms of the LSTM-FCN is only being applied on a single time step. This attributes to why LSTM with dimension shuffle by itself performs poorly. However, as noticed in Section V-C, when applying the LSTM block with dimension shuffle and the FCN block parallelly, the blocks augment each other, while improving its overall performance. To the best of our knowledge, we believe the LSTM block with a dimension shuffle acts as a regularizer to the FCN block, forcing the FCN block to improve its performance.

E. REPLACING LSTM WITH GRU, RNN, AND A DENSE LAYER

Since the usage of the LSTM block when applying dimension shuffle to the input is atypical, we replace the LSTM block with a GRU block (8, 64, 128 cells), basic RNN block (8, 64, 128 cells), and a Dense block with a sigmoid activation function (8, 64, 128 units) on all 128 datasets (total of 384 experiments on each model).The intuition behind selecting an RNN block and a GRU block is that these blocks have similar properties to an LSTM block, and differ only in their capacity to learn long term temporal dependencies. Furthermore, a dense layer is selected to compare against the atypical usage of the LSTM block, so that we may analyze whether the complex interaction within the recurrent gates of the LSTM can be simplified into a single fully connected layer. We chose the sigmoid activation function for the Dense block, instead of the standard Rectifying Linear Unit (ReLU) activation, as we wish to compare the effectiveness of the

TABLE 4. Ablation test - Wilcoxson signed-rank test comparing LSTM-FCN with GRU-FCN, RNN-FCN, and Dense-FCN. The values in parenthesis depicts the number of wins, ties, and losses the row index has with the header. Red cell depicts when the test fails to reject the null hypothesis.

	GRU-FCN	RNN-FCN	Dense-FCN
LSTM-FCN	6.33E-16 (243/38/103)	1.34E-17 (247/41/96)	2.81E-10 (231/35/118)
GRU-FCN		1.05E-02 (185/53/146)	1.57E-01 (160/49/175)
RNN-FCN			7.55E-05 (135/49/200)

gating effect exhibited by the 3 gates of the LSTM. The majority of the gates of the LSTM use the sigmoid activation function. Therefore, we construct the Dense block to also use the same. The input to the GRU block, RNN block, and Dense block had a dimension shuffle applied onto it. Replacing the LSTM block of LSTM-FCN with a GRU block was first proposed by Elsayed *et al.* [47]. Table 4 summarizes a Wilcoxson signed-rank test when LSTM-FCN with dimension shuffle is compared to GRU-FCN, RNN-FCN, and Dense-FCN.

The Wilcoxson signed-rank test depicts LSTM-FCN with dimension shuffle to statistically outperform GRU-FCN, RNN-FCN, Dense-FCN. Surprisingly, the model to perform most similar to LSTM-FCN with dimension shuffle is Dense-FCN. LSTM-FCN outperforms Dense-FCN in 231 experiments, ties in 35 experiments and performs worse in 118 experiments.

An interesting observation is that GRU-FCN does not statistically outperform Dense-FCN. Based on our 384 experiments, GRU-FCN outpeforms Dense-FCN in 160 experiments, ties in 49 experiments, while performing worse in 175 experiments. As a disclaimer, we performed each of these experiments only once, therefore there may be some deviation when run multiple times due to the inherent variance of training using random initialization. However, due to the sample size of 384, we believe the variance will not be significant to result in a different conclusion.

VI. CONCLUSION & FUTURE WORK

In this paper, we provide a better understanding of LSTM-FCN, ALSTM-FCN and their sub-modules through a series of ablation tests (3627 experiments). We show that *z-normalizing* the whole dataset yields to results different than *z-normalizing* each sample. For the model LSTM-FCN and ALSTM-FCN, we recommend *z-normalizing* the whole dataset only in situations when it is known that the training set is a good representation of the global population. Moreover, our ablation tests show that the LSTM/ALSTM block and the FCN block yields to a better performing model when applied in a conjoined manner. Further, the performance of LSTM-FCN is enhanced only when dimension shuffle is applied before the LSTM block. Finally, in this paper, we substitute the LSTM block with either a GRU block, a RNN block or a Dense block to observe the effect of such a substitution. Our results indicate LSTM-FCN to outperform GRU-FCN, RNN-FCN and Dense-FCN.

An exciting area for future work is to investigate why LSTM-FCN and ALSTM-FCN underperform in a few UCR datasets and to ascertain whether the models can be made more robust to the various types of time series data. Furthermore, integrating the models in both low-power systems and wearables for on-device classification is of great interest. Finally, further inroads can be made in streaming time series classification by the utilization of these models.In the future, researchers that want to implement deep learning models for time series classification need to focus on generalization of the model on unseen sequences, and reduce overfitting as the UCR repository contain small real world data sets.

ACKNOWLEDGMENT

The authors would like to thank all the researchers that helped create and clean the data available in the updated UCR Time Series Classification Archive. They would also like to show their gratitude to the administrators of the UCR Time Series Classification Archive, Dau et al. Sustained research in this domain would be much more challenging without their efforts.

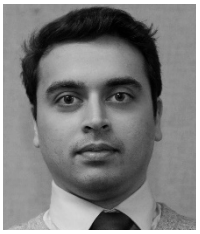
Further, the authors would like to acknowledge the Research Open Access Publishing (ROAAP) Fund of the University of Illinois at Chicago for financial support towards the open access publishing fee for this article.

(Fazle Karim and Somshubra Majumdar contributed equally to this work.)

REFERENCES

[1] H. A. Dau *et al.* (2018). "The UCR time series archive." [Online]. Available: https://arxiv.org/abs/1810.07758
[2] A. Sharabiani, H. Darabi, A. Rezaei, S. Harford, H. Johnson, and F. Karim, "Efficient classification of long time series by 3-d dynamic time warping," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 10, pp. 2688–2703, Oct. 2017.
[3] A. Sharabiani, F. Karim, A. Sharabiani, M. Atanasov, and H. Darabi, "An enhanced Bayesian network model for prediction of students' academic performance in engineering programs," in *Proc. IEEE Global Eng. Educ. Conf. (EDUCON)*, Apr. 2014, pp. 832–837.
[4] F. Karim, S. Majumdar, H. Darabi, and S. Harford. (2018). "Multivariate LSTM-FCNs for time series classification." [Online]. Available: https://arxiv.org/abs/1801.04503
[5] L. Wei and E. Keogh, "Semi-supervised time series classification," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2006, pp. 748–753.
[6] J. W. Taylor, P. E. McSharry, and R. Buizza, "Wind power density forecasting using ensemble predictions and time series models," *IEEE Trans. Sustain. Energy*, vol. 24, no. 3, pp. 775–782, Sep. 2009.
[7] R. S. Tsay, *Analysis of Financial Time Series*, vol. 543. Hoboken, NJ, USA: Wiley, 2005.
[8] K. Sternickel, "Automatic pattern recognition in ECG time series," *Comput. Methods Programs Biomed.*, vol. 68, no. 2, pp. 109–115, 2002.
[9] J. Theiler, S. Eubank, A. Longtin, B. Galdrikian, and J. D. Farmer, "Testing for nonlinearity in time series: The method of surrogate data," *Phys. D, Nonlinear Phenomena*, vol. 58, nos. 1–4, pp. 77–94, 1992.
[10] E. A. Maharaj, P. D'Urso, and J. Caiado, *Time Series Clustering and Classification*. Boca Raton, FL, USA: CRC Press, 2019.
[11] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks: A unified approach to action segmentation," in *Computer Vision* (Lecture Notes in Computer Science). Amsterdam, The Netherlands: Springer, 2016, pp. 47–54.
[12] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 1033–1040.
[13] B. J. Jain and D. Schultz, "Asymmetric learning vector quantization for efficient nearest neighbor classification in dynamic time warping spaces," *Pattern Recognit.*, vol. 76, pp. 349–366, Apr. 2018.
[14] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases," in *Proc. Int. Conf. Found. Data Org. Algorithms*. Berlin, Germany: Springer, 1993, pp. 69–84.

[15] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Data Mining Knowl. Discovery*, vol. 7, no. 4, pp. 349–371, Oct. 2003.
[16] Y. Chen *et al.* (Jul. 2015). *The UCR Time Series Classification Archive*. [Online]. Available: https://www.cs.ucr.edu/~eamonn/time_series_data/
[17] H. A. Dau *et al.* (Oct. 2018). *The UCR Time Series Classification Archive*. [Online]. Available: https://www.cs.ucr.edu/~eamonn/time_series_data_2018/
[18] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: A novel symbolic representation of time series," *Data Mining Knowl. Discovery*, vol. 15, no. 2, pp. 107–144, Apr. 2007.
[19] M. G. Baydogan, G. Runger, and E. Tuv, "A bag-of-features framework to classify time series," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2796–2802, Nov. 2013.
[20] P. Schäfer, "The BOSS is concerned with time series classification in the presence of noise," *Data Mining Knowl. Discovery*, vol. 29, no. 6, pp. 1505–1530, Sep. 2014.
[21] P. Schäfer, "Scalable time series classification," *Data Mining Knowl. Discovery*, vol. 30, no. 5, pp. 1273–1298, 2016.
[22] P. Schäfer and U. Leser. (2017). "Fast and accurate time series classification with WEASEL." [Online]. Available: https://arxiv.org/abs/1701.07681
[23] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Mining Knowl. Discovery*, vol. 29, no. 3, pp. 565–592, Jun. 2014.
[24] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time-series classification with COTE: The collective of transformation-based ensembles," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 9, pp. 2522–2535, Sep. 2015.
[25] Z. Cui, W. Chen, and Y. Chen. (2016). "Multi-scale convolutional neural networks for time series classification." [Online]. Available: https://arxiv.org/abs/1603.06995
[26] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 1578–1585.
[27] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM fully convolutional networks for time series classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2017.
[28] Y. Kim, J. Sa, Y. Chung, D. Park, and S. Lee, "Resource-efficient pet dog sound events classification using LSTM-FCN based on time-series data," *Sensors*, vol. 18, no. 11, p. 4019, 2018.
[29] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[30] L. Trottier, P. Giguère, and B. Chaib-Draa. (May 2016). "Parametric exponential linear unit for deep convolutional neural networks." [Online]. Available: https://arxiv.org/abs/1605.09332
[31] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. (2013). "How to construct deep recurrent neural networks." [Online]. Available: https://arxiv.org/abs/1312.6026
[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[33] A. Graves *et al.*, *Supervised Sequence Labelling with Recurrent Neural Networks*, vol. 385. Springer, 2012.
[34] K. Cho *et al.* (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation." [Online]. Available: https://arxiv.org/abs/1406.1078
[35] G. King and L. Zeng, "Logistic regression in rare events data," *Political Anal.*, vol. 9, no. 2, pp. 137–163, May 2001.
[36] F. Chollet *et al.* (2015). *Keras*. [Online]. Available: https://github.com/fchollet/keras
[37] M. Abadi *et al.*. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: https://www.tensorflow.org/
[38] D. P. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980
[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
[40] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1026–1034.
[41] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
[42] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. AAAI*, vol. 4, 2017, p. 12.
[43] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.
[44] F. Wilcoxon and R. A. Wilcox, *Some Rapid Approximate Statistical Procedures*. Wayne, NJ, USA: Lederle Laboratories, 1964.
[45] H. Abdi, "The Bonferonni and Šidák corrections for multiple comparisons," *Encyclopedia of Measurement and Statistics*, vol. 3, 1st ed. Newbury Park, CA, USA: SAGE, 2007, pp. 103–107.
[46] P. Schäfer and U. Leser. (2017). "Fast and accurate time series classification with WEASEL." [Online]. Available: https://arxiv.org/abs/1701.07681
[47] N. Elsayed, A. S. Maida, and M. Bayoumi. (2018). "Deep gated recurrent and convolutional network hybrid model for univariate time series classification." [Online]. Available: https://arxiv.org/abs/1812.07683



FAZLE KARIM received the B.Sc. degree in industrial engineering from the University of Illinois at Urbana–Champaign, in 2012, and the M.Sc. degree in industrial engineering from the University of Illinois at Chicago, in 2016, where he is currently pursuing the Ph.D. degree with the Mechanical and Industrial Engineering Department.

He is also a Researcher with the Prominent Laboratory, the university's foremost research facility in process mining. His current research interests include education data mining, health care data mining, time series analysis, and adversarial attacks.

SOMSHUBRA MAJUMDAR received the B.S. degree in computer engineering from the University of Mumbai, in 2016. He is currently pursuing the M.S. degree in computer science from the University of Illinois at Chicago. He is also an aspiring artificial intelligence Researcher with the University of Mumbai. His current research interests include the domain of image classification and segmentation using convolutional neural networks, time series classification using recurrent neural networks, and machine learning.

HOUSHANG DARABI (S'98–A'00–M'10–SM'14) received the Ph.D. degree in industrial and systems engineering from Rutgers University, New Brunswick, NJ, USA, in 2000.

He is currently an Associate Professor with the Department of Mechanical and Industrial Engineering, University of Illinois at Chicago (UIC), and also an Associate Professor with the Department of Computer Science, UIC. He has been a contributing author of two books in the areas of scalable enterprise systems and reconfigurable discrete event systems. His research has been supported by several federal and private agencies, such as the National Science Foundation, the National Institute of Standard and Technology, the Department of Energy, and Motorola. He has extensively published on various automation and project management subjects, including wireless sensory networks for location sensing, planning and management of projects with tasks requiring multi-mode resources, and workflow modeling and management. He has published in different prestigious journals and conference proceedings, such as the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, and *Information Sciences*. His current research interests include the application of data mining, process mining, and optimization in design and analysis of manufacturing, business, project management, and workflow management systems.